Artificial Intelligence in Medicine. Volume 107, 2020, 101924. https://doi.org/10.1016/j.artmed.2020.101924

Detection of early stages of Alzheimer's disease based on MEG activity with a randomized convolutional neural network

Manuel Lopez-Martin, Angel Nevado, Belen Carro

Abstract— The early detection of Alzheimer's disease can potentially make eventual treatments more effective. This work presents a deep learning model to detect early symptoms of Alzheimer's disease using synchronization measures obtained with magnetoencephalography. The proposed model is a novel deep learning architecture based on an ensemble of randomized blocks formed by a sequence of 2D-convolutional, batch-normalization and pooling layers. An important challenge is to avoid overfitting, as the number of features is very high (25755) compared to the number of samples (132 patients). To address this issue the model uses an ensemble of identical sub-models all sharing weights, with a final stage that performs an average across sub-models. To facilitate the exploration of the feature space, each sub-model receives a random permutation of features. The features correspond to magnetic signals reflecting neural activity and are arranged in a matrix structure interpreted as a 2D image that is processed by 2D convolutional networks.

The proposed detection model is a binary classifier (disease/non-disease), which compared to other deep learning architectures and classic machine learning classifiers, such as random forest and support vector machine, obtains the best classification performance results with an average F1-score of 0.92. To perform the comparison a strict validation procedure is proposed, and a thorough study of results is provided.

Index Terms— Alzheimer's disease detection; deep learning; convolutional neural network; ensemble model; magnetoencephalography

* Correspondence: mlopezm@acm.org; Tel.: +34-983-423-980, Fax: +34-983-423-667
The authors declare that there is no conflict of interest regarding the publication of this paper

1. INTRODUCTION

The early detection of Alzheimer's disease (AD) is a critical medical and social challenge. There have been many approaches to this challenge using machine learning (ML) [1], including current advancements with deep learning [2]. In this paper we present a novel deep learning model for early detection of AD using magnetoencephalography (MEG) signals obtained from patients with Mild Cognitive Impairment (MCI). MCI [3] is a stage between the expected cognitive decline of healthy aging and dementia. Patients with MCI are at an increased risk of developing Alzheimer's Disease [4].

Magnetoencephalography (MEG) has been shown to have good sensitivity to detect AD in its early stages [5]. Previous work has applied Machine Learning (ML) to MEG data for this purpose [1,6,7]. ML models have also been applied to detect AD using other neuroimaging signals such as functional magnetic resonance imaging (fMRI) [8,9], magnetic resonance imaging (MRI) [2] and electroencephalography (EEG) [10,11,12]. Similarly, neuroimaging data have been analyzed with machine learning techniques to investigate a large variety of brain conditions and processes including rapid eye movement disorders [11], speech tasks classification [13], Parkinson's disease [14], epilepsy [15], spinal cord injury [15], scene representations [16] and ocular and cardiac artifacts detection [17]. The ML models that have been employed in this context are: (1) classic models - Random Forest, Bayesian Network, Decision trees, K-Nearest Neighbors, Logistic Regression, Support Vector Machines and Linear discriminant

M. Lopez and B. Carro are with the Dpto. TSyCeIT, ETSIT, Universidad de Valladolid, Paseo de Belén 15, Valladolid 47011, Spain; (mlopezm@acm.org).
A. Nevado is with the Laboratory for Cognitive and Computational Neuroscience, Center for Biomedical Technology, Campus Montegancedo, Pozuelo de Alarcón, Madrid 28223, Spain and the Experimental Psychology Department of the Complutense University of Madrid.

analysis - [6,7], (2) deep learning models - Convolutional neural networks (CNN) [2,9,11,14,15], Recurrent neural networks (RNN) [10] and custom architectures [8,13,17,18] - (3) Bayesian ML models [12].

In this work we propose a novel architecture for the detection of Mild Cognitive Impairment (MCI). The architecture is based on an ensemble of randomized learning blocks, each consisting of a sequence of 2D-convolutional layers [19], batch-normalization layers [20] and max-pooling layers [21]. Batch-normalization layers facilitate convergence during training and provide a regularization effect that helps prevent overfitting. Max-pooling layers reduce the number of parameters (weights) of the network, which reduces the possibility of overfitting. A randomization is applied to the input of the different blocks, allowing for maximum exploration of the interaction between features, which is important considering the local-based nature of the convolutional networks. A weight sharing strategy between blocks is used to keep the complexity of the final model low, given the small number of training samples. To compare the detection performance of the proposed model we compare it with several classic ML models: Logistic Regression (LR), Naïve Bayes (NB), Gradient Boosting Machine (GBM), Random Forest (RF), Gaussian Process Classifier (GP Classifier) and Support Vector Machine (SVM) as well as more advanced deep learning models based on convolutional neural networks (CNN) [19] to classify individuals as healthy aging subjects or MCI patients.

The features used to train the different models are based on a measure of statistical dependence, mutual information, between the MEG signals of pairs of sensors, with a total of 102 sensors and 5151 pairs of sensors per subject. After post-processing the signals, we obtain five final features for each pair of sensors. The total number of individuals for the experiment is 132 (59% with MCI), and the total number of features, after post-processing of the MEG signals, is 25755 per individual.

The high ratio of number of features, 25755, to number of samples, 132, is associated with a high risk of overfitting. We have, therefore, taken special care to guarantee the generalization of results. Deep learning (DL) models are known to need large datasets with a large number of samples. We have circumvented this by using Deep Learning architectures, based on Convolutional Neural Networks, with a very limited number of parameters by carefully selecting the filter shapes, max-pooling layers and employing weight-sharing for the more complex architectures.

An exhaustive comparison of the results is provided for all models. Special care has been taken in the selection of classification performance metrics and the test strategy, which is based on a nested cross-validation approach. Considering the small number of samples, it is especially important to guarantee the distribution of the performance metrics. For this reason, the mean, standard deviation, maximum and minimum values and the 25%, 50% (median) and 75% quartiles are provided for all performance metrics and all models. The selected classification performance metrics are accuracy, F1-score, precision and recall. These four metrics provide a comprehensive description of the different aspects and objectives of a classifier. We need to mention the importance of the recall metric for this particular classification problem, since it is essential to avoid false negatives (subjects for which MCI is not detected).

To apply a 2D convolutional network to a one-dimensional vector of features it is necessary to reshape the features into a matrix structure (pseudo-image) with an arbitrary allocation of features to positions within the new matrix. This approach of transforming one-dimensional features into a two-dimensional structure, to be used by a 2D-CNN, has already been applied in other areas with excellent results [22,23]. In this work, we extend this approach by creating an ensemble of CNN models, each with different inputs formed by matrices created by randomly permuting the original features (from the one-dimensional vector of features) to different positions within each matrix. The output of each CNN model is averaged forming the output of the ensemble. The ensemble has an additional regularization effect that is very helpful in addressing this classification problem.

To understand why the randomized 2D-CNN model yields good results we have to take into that we are dealing with high-dimensional noisy signals of which only a small number of samples are available. This creates a scenario prone to overfitting. The use of stochastic methods to combat overfitting, as considered in many previous works [24,25,26], was the basis to apply randomization to the input part of the network [26] instead of the intermediate [25] or final [24] parts. In addition, the fact that CNN models exploit spatial local correlations was a motivation to introduce the random permutation of features, with the goal of increasing the exploration of combination of variables that were not close to each other in their original locations.

The contributions of this work are:

- To introduce a novel deep learning architecture based on Convolutional Neural Networks for the diagnosis of MCI, a prior stage of Alzheimer's disease.
- To compare the proposed model with other CNN architectures and classic ML models to detect MCI using a large collection of features obtained from MEG signals
- To handle noisy and scarce training data with a 2D-CNN ensemble model that averages a collection of sub-models based on the stochastic permutation of the input.

- To build a model with excellent classification results in a challenging scenario (scarce samples with a large feature's dimensionality).

The paper is organized as follows: Section 2 summarizes previous works. Section 3 describes the dataset and models employed. Section 4 details the results and section 5 presents the conclusions.

2. RELATED WORKS

The approach adopted in this work, which consists in the transformation of MEG signals into images to subsequently apply CNN models (which are especially suitable to handle images) has already been explored in other works investigating brain function. In [15] a CNN to classify patients with epilepsy and spinal cord injury was proposed. They calculated relative power spectra from MEG signals. The activity was presented in a channel vs. time matrix and treated as an image by a CNN network. A similar approach was adopted in [13], this time using the raw data from time-series recordings, with a previous feature extraction based on Independent Component Analysis (ICA). The objective of this last work was to predict the age of children based on various speech tasks. This approach is different from using anatomical brain images - to identify brain conditions, such as using MRI images [2]. The identification of a data matrix as an image to allow the subsequent application of a CNN has also been explored in other fields, such as network intrusion detection and multimedia quality of experience prediction [22,23], providing, in both cases, an excellent improvement in classification results.

The approach taken in [27] employs a CNN architecture that is used as a feature extraction tool in conjunction with FreeSurfer [27]. The work proposes an MCI detector using magnetic resonance imaging (MRI) data. Features extracted from MRI are further processed with PCA (dimensionality reduction) and Lasso (feature selection). The final processed features are used by an Extreme Learning Machine for MCI classification.

Randomization of the input features has also been applied to AD detection in an ensemble configuration of neural networks (NN), with the NNs trained with randomly selected samples and features obtained after calculating the Pearson correlation coefficient between the fMRI time-series of pairs of voxels [8]. The randomization of the input features or the intermediate network representations has also been applied in other areas. In [28] an image classifier with a low computational infrastructure was proposed. This classifier employed group convolutions followed by a random combination of intermediate outputs. A different image classifier was presented in [29], where a CNN network was replaced by a series of linear classifiers that were fed by small portions of the image with a final aggregation (averaging) per class. In this case, there was no randomization of the input. Instead, input segregation was employed, followed by separate classifiers with a final aggregation of results. Still in the field of image classification, [30] proposed a simple architecture based on the application of several fixed random perturbations to the original image with a final linear combination of the perturbed intermediate images after a rectified linear unit (ReLU) activation function. Surprisingly, the resulting architecture produces similar classification results to a CNN network. In [31], an input shifting is performed within the CNN kernels rather than the inputs. The work in [32] points out the effect of the locality property for images and how when this property is lost (by randomly mixing pixels) the usual performance of classic CNN is greatly affected, while other CNN architectures (such as dilated convolutions) can provide much better results by integrating non-adjacent pixels into the same receptive field, which is also the purpose, by other means, of randomizing the input positions. Applying a series of dilated convolutions with different dilation rates provides an effect similar to input randomization, in a more deterministic way and also requiring a deeper network.

The work presented here differs from the works mentioned above in that it also employs randomization of the input, but as a means to create an extended dataset by permuting the inputs of the original dataset (treated as a pseudo-image), producing several shuffled pseudo-images that are handled by separate networks with a final aggregated result. This ensemble-like architecture produces better results than individual CNN networks, and is more robust against overfitting, which is a real problem considering the small number of samples and the large number of features in this type of AD detection tasks.

3 METHODS DESCRIPTION

In this Section we provide details of the dataset used to carry on the experiments, the architecture proposed to detect Mild Cognitive Impairment and the validation/test strategy applied. The employed dataset and the proposed model are presented on sections 3.1 and 3.2 respectively. The validation/test strategy is provided in section 3.3.

3.1 Selected dataset

Data was acquired in the Centre for Biomedical Technology, (Madrid, Spain). Resting-state data was collected from 78 Mild Cognitive Impairment (MCI) patients and 54 healthy aging participants. The demographic data is provided in Table I.

Diagnostic Group	Age	MMSE	Sex (% Female)
Healthy Aging Participants	69.95±4.40 [61-80]	29.33±0.79 [27-30]	72.73
MCI Patients	74.02±6.40 [58-87]	27.61±2.17 [22-30]	51.16

Table I. Mean ± standard deviation and range (between brackets) of the age and Mini Mental State Exam (MMSE) for the two diagnostic groups, and proportion of males and females.

MCI patients fulfilled the following criteria [33]: (i) memory complaint, corroborated by an informant, (ii) objective memory impairment for age, (iii) relatively preserved general cognition for age, (iv) essentially intact activities of daily living, and (v) not meeting the criteria for dementia. Exclusion criteria were having a neurological disease, a psychiatric disorder, or any severe illness. The ethics committee at the MEG center approved the study and all participants gave written informed consent.

Participants sat in a 306-channel Elekta MEG system (Elekta Oy, Helsinki, Finland) during 4 minutes with eyes closed. The sampling rate was 1000 Hz. A temporal signal space separation algorithm (tSSS) was applied to reduce environmental and biological noise components [34]. MEG data preprocessing was performed with FieldTrip [35]. Recordings were band-pass filtered between 2 and 45 Hz. From the available sensors, the analysis focused on the 102 magnetometers. The time-series were subsequently segmented into 2 second epochs. Ocular, muscular and jump artifacts were automatically detected with Fieldtrip and the corresponding segments discarded from the analysis.

The statistical dependencies between the signal time-series $x(t)$ from sensor i and $y(t)$ from sensor j were measured with mutual information:

$$I(X;Y) = \sum_{y\in Y} \sum_{x\in X} p_{(X,Y)}(x,y) log\left(\frac{p_{(X,Y)}(x,y)}{p_X(x)p_Y(y)}\right) \qquad (1)$$

Where $p(X,Y)$ is the joint probability of $x(t)$ and $y(t)$, were value pairs were obtained from the same time instant t, and p_x and p_y are the marginal probabilities of $x(t)$ and $y(t)$. Prior to the calculation of the probability distributions the time series were discretized into 15 bins to yield the value ranges X and Y. Subsequently, a normalized mutual information value was obtained:

$$I_{norm}(X;Y) = \frac{I(X;Y)}{\sqrt{H(X;X)H(Y;Y)}} \qquad (2)$$

where $H(X;X)$ and $H(Y;Y)$ are the entropies of X and Y respectively

$$H(Z;Z) = -\sum_{z\in Z} p_Z(z) log(p_Z(z)) \qquad (3)$$

A value of normalized mutual information was obtained for each participant, pair of sensors and epoch. Next, five summary values were derived by collapsing values across epochs: mean, median, standard deviation, mean absolute deviation and range (maximum-minimum).

The total number of features (25755) corresponds to the number of ways to select 2 sensors out of 102 sensors multiplied by the 5 different summary statistics of the mutual entropy per pair of MEG signals (with one MEG signal per sensor). Feature values are continuous in the range [0-1]. Fig. 1 shows a diagram of the process followed to obtain the features for each patient.

Considering the small number of samples and the large number of features it was necessary to perform feature selection with the aim to achieve an ideal ratio of number of samples to number of features of 10. To obtain the final set of features we applied three features selection techniques to the dataset: (1) Random Forest variable importance [36], (2) selection based on iterative feature elimination after comparing with random features created by shuffling the original features (Boruta) [37] and (3) recursive feature elimination (RFE) (caret) [36]. All these methods provide a ranking of features from highest to lowest importance. The features resulting from the intersection of the feature sets provided by these three selection techniques were arranged into two final set of features: one set with 870 features and other with 20 features (the best ranked features in the intersection set). These feature sets 1 and 2 (FS1 and FS2) corresponding to the previously defined 20 and 870 features, respectively, were then used by the different algorithms presented in this work. Which feature set was used in each algorithm is specified in section 4 (Results).

The aforementioned scheme for performing feature selection attempts to avoid the selection bias problem[38].This problem happens when performing feature selection with a reduced number of samples and a large number of features. This problem is due to random correlations between the features and the outcomes, even for unimportant features. The intersection between the best sets of three different selection techniques is intended to avoid this problem. In addition, two of the R packages used for feature selection: caret and Boruta [36,37], both take into account the selection bias problem. In caret [36], the feature selection method incorporates the resampling solution proposed in [38] using cross validation within each iteration of the feature selection process. Boruta [37] implements a sophisticated selection technique based on comparing each feature with a synthetic shadow feature made by shuffling the values of the original feature.

The number of 20 features in FS1, was selected after some ad hoc tests, as a good compromise between a reduced number of features and the classification performance obtained. Similarly, a number of 870 was chosen for FS2 as it can be factored into two similar numbers, namely 29 and 30, allowing the construction of a squared-shaped matrix as input for the 2D-CNN models.

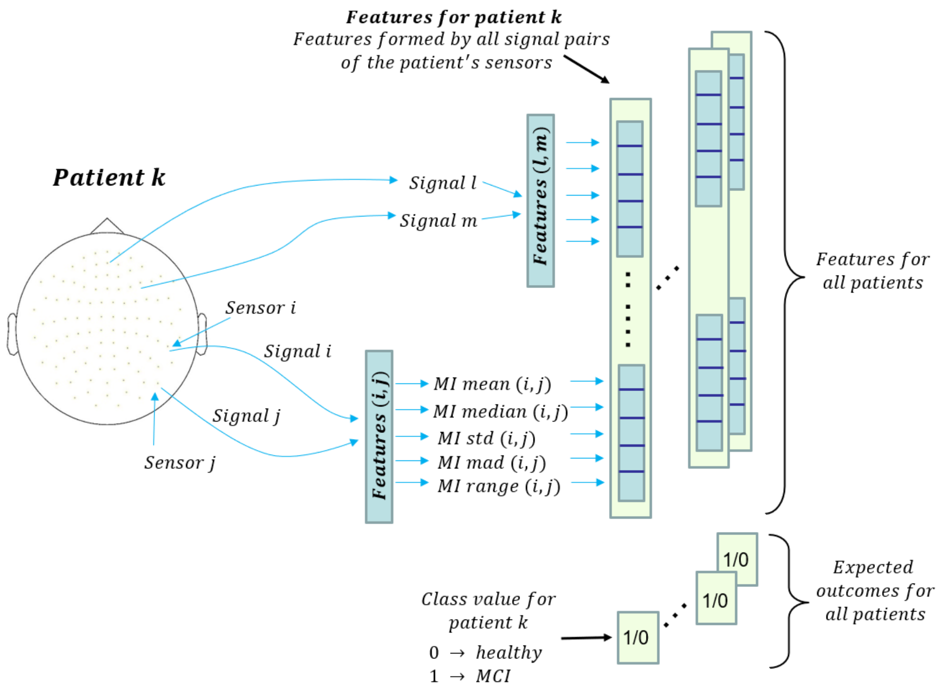


Fig. 1. Diagram of the generation of features associated with a generic patient (patient k). Each pair of patient sensors produces five features. The diagram shows two of these pairs. The sensor layout on the left was created with Fieldtrip [35]

3.2 Model description

The objective is to explore and compare several CNN architectures to detect Mild Cognitive Impairment using a large collection of features obtained from MEG signals while patients are at rest. We investigated the following CNN architectures:

- 1-D CNN: A classic 1-D CNN [39] applied to the one-dimensional vector of features derived from the MEG signals.
- 1-D CNN + Gaussian Process Classifier: Similar to the previous architecture with a final layer formed by a Gaussian Process Classifier [40].
- 2-D CNN: A classic 2-D CNN [19] applied to synthetic images formed by an arbitrary arrangement of the features obtained from the MEG signals.
- Randomized 2-D model: This is the novel architecture proposed for this work achieving the best classification results. It consists on an ensemble of 2D-CNN networks with a permutation of the input values for each network.

The dataset used to train the models was FS2 with 870 features for the 2-D CNNs and FS1 with 20 features for de 1D models. Considering the small number of samples and the large number of features, the application of a CNN poses difficulties since they typically require many samples. The number of parameters (weights) of a CNN is usually large, which implies that many samples are necessary to tune the parameters and avoid overfitting. Therefore, special care has been taken to create CNN architectures with a small number of weights and to ensure the generalization of the results to the greatest extent possible.

Starting with the 1-D CNN [39] architecture, Fig. 2 presents the 1-D CNN model. It is a simple model with just one 1-D CNN layer (5 filters, kernel size of 5 and stride of 5) followed by two fully connected layers (FCL) with 5 and 2 nodes respectively. Between the CNN and FCL layers it is necessary to perform a vector flattening of the output tensor of the CNN layer, generating a vector of 20 elements that is the input to the FCL layers. The activation function of the layers is ReLU with a final softmax activation for the last layer and cross entropy as the loss function. The last layer has a dropout [24] with a drop rate of 0.5. The total number of trainable weights is 147 with a batch size of 10 and 80 epochs of training with an early stopping of 30 epochs. The 1-D CNN model shown in Fig. 2 is the 1-D CNN model that presents the best results after testing several variants that included additional 1-D CNN layers, batch-normalization and max-pooling layers. The 1-D CNN model has 147 trainable parameters (weights).

The training samples used for the 1-D CNN models are vectors of 20 features obtained after a feature selection process (section 3.1) from the 25755 original features. This model is applied to a 1-D vector of features. The reason for using the FS1 dataset (20 features) is due to the best results obtained compared to the FS2 dataset (870 features) when we apply the 1D-CNN model.

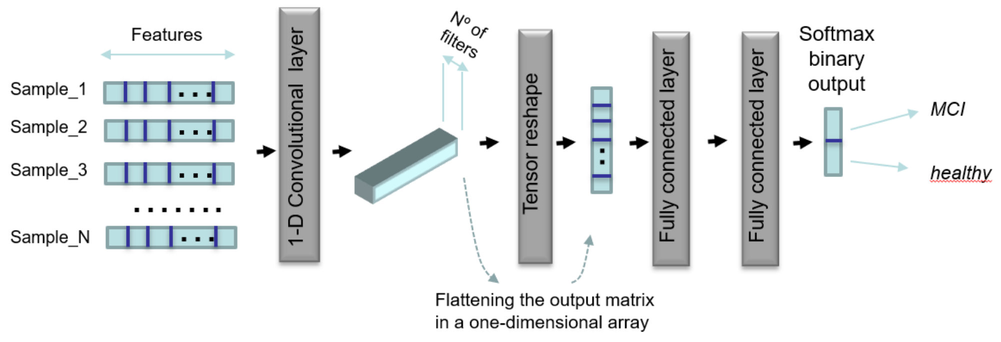


Fig. 2. 1-D CNN model

The model in Fig. 3 is similar to the model in Fig. 2 with the addition of a gaussian process classifier (GPC) [40] to the output of the last layer of the 1-D CNN model. A GPC is a non-parametric bayesian model, well known for its suitability to classify with a small number of features. Nevertheless, this model did not produce better results than the simpler 1-D CNN model. To train the 1-D CNN+GPC model, we used a two-step process, with a previous training phase of the 1-D CNN model and a subsequent and independent training phase of the GPC model, using as inputs the outputs from the last layer of the already trained 1-D CNN model. The two-phase training could be one of the reasons for the lower performance of this model compared to the 1-D CNN model, since we did not perform an end-to-end training of the complete model. The kernel used for the GPC was an RBF kernel. The 1-D CNN+GPC model has 140 trainable parameters (weights).

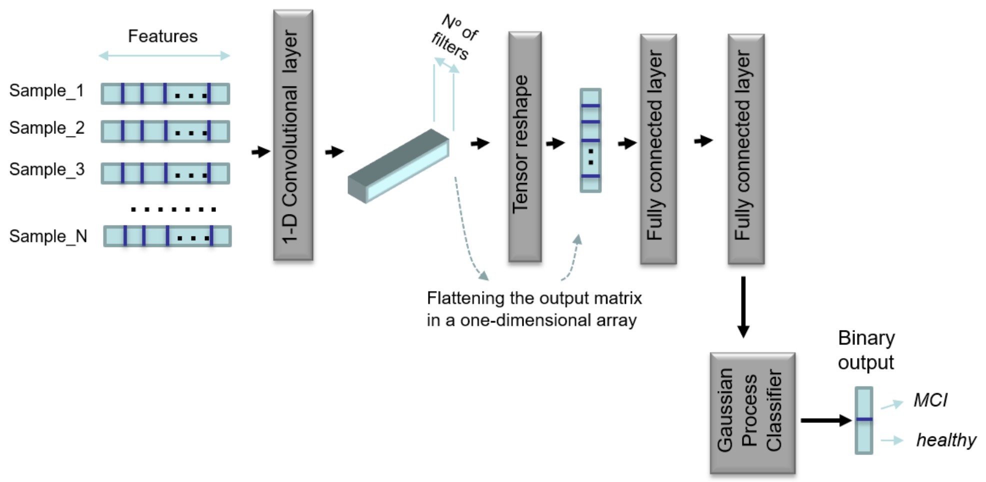


Fig. 3. 1-D CNN + GPC model

The model in Fig. 4 corresponds to a 2-D CNN [19]. In this case the 1-D feature vector of 870 features (section 3.1) was reshaped into a matrix structure (pseudo-image) of dimension 30 x 29. The arrangement of the features in a pseudo-image allows to apply a 2-D CNN as shown in Fig. 4. Several architectures were tested. The architecture presented in Fig. 4 is the one that provides the best results for this type of models, after testing several variants that included additional 2-D CNN layers, batch-normalization and max-pooling layers. This architecture is formed by three 2-D CNN layers with intermixed batch-normalization [20] and max-pooling [21] layers. The last model components are 2 FCL (Fig. 4) with 5 and 2 nodes respectively. The configuration of the three 2D-CNN layers is, in order, as follows: first (2 filters, kernel size of 3 and stride of 2), second (3 filters, kernel size of 3 and stride of 1) and third (4 filters, kernel size of 2 and stride of 1). Between the CNN and FCL layers it is necessary to perform a vector flattening of the output tensor of the last CNN layer, generating a vector of 16 elements, which is the input to the FCL layers. The 2 max-pooling layers have a pool size of 2. The activation function of the layers is ReLU with a final softmax activation for the last layer and cross entropy as the loss function. The last layer has a dropout [24] with a drop rate of 0.5. The total number of trainable weights is 236, with a batch size of 10 and 80 epochs of training, with an early stopping of 30 epochs. The 2-D CNN model has 236 trainable parameters (weights) and 10 non-trainable parameters, associated to batch-normalization layers.

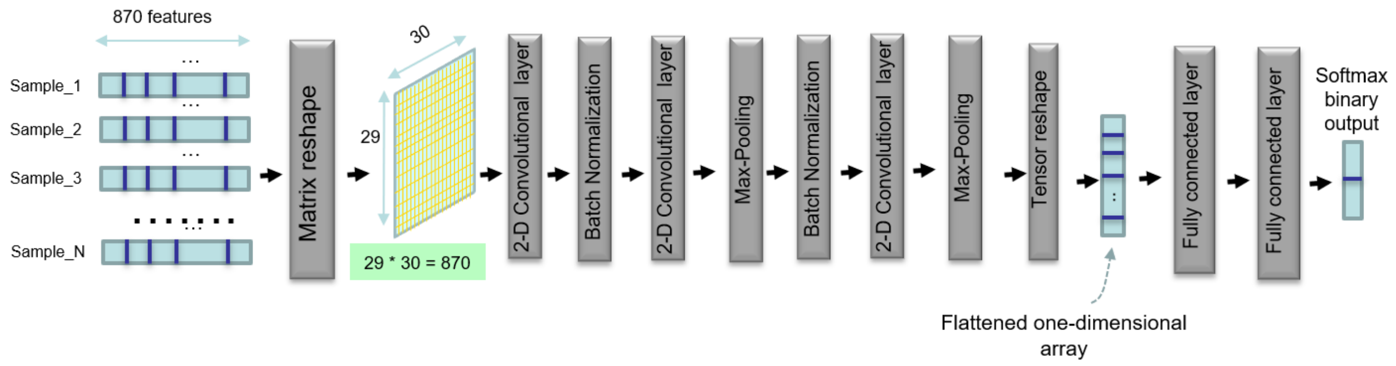


Fig. 4. 2-D CNN model

The last model is presented in Fig. 5. This is the model that showed the best classification results (section 4). This model is formed by a set of k sub-models, each identical to the model presented in Fig. 4. The input to each of the k sub-models is formed by the same pseudo-image matrix structure used for the model in Fig. 4, with a previous additional random permutation of the columns (features). The permutation of the columns is applied independently for each of the k inputs of the model. Therefore, the process followed to create the k inputs involves performing k random permutations of all the columns of the original dataset, producing k column-permuted datasets. The random permutations, once defined, are maintained throughout the training and prediction phases i.e. the position for each feature in each of the k inputs is different and established randomly, but once they are established, their position is maintained, otherwise training and prediction would be completely stochastic and impossible. These datasets are the source for generating k pseudo-images for the k sub-models. Considering the large number of weights of the final model, due to the repetition of sub-models, we have implemented a weight sharing among all sub-models, keeping the total number of trainable weights at 236, which is important to reduce overfitting. To incorporate the outputs of all the models, we used an element-wise average operation between the k flattened output vectors produced by the k sub-models. The resulting vector is used as input for the last FCL layers. After testing several values of k, we chose a value of 10, which provides sufficient input randomization while keeping the computational cost under reasonable limits. The randomized 2-D CNN model has 236 trainable parameters and 10 non-trainable parameters (for batchnormalization layers). This is the same number of parameters (weights) as the 2-D CNN model. This is a desired result since we achieve a much versatile architecture with the same model complexity thanks to the weight sharing and the final average function.

It is important to note that our initial dataset consists of 25755 features that correspond to the number of ways to select 2 sensors out of 102 sensors multiplied by the 5 different summary statistics per MEG signal pair. Therefore, the location of these features in the dataset does not follow a clear order (under a distance-metric), nor does it represent an easy association with the location of the sensors (Fig. 1). Additionally, the difficulty in creating location-aware features increases with the pre-processing (feature selection) applied to the dataset (Section 3.1). In summary, we have a large number of features with a non-informative position. CNN models exploit the correlation between adjacent features using a location-based metric. Its application to image datasets is fully justified, as smooth changes in pixel properties in local proximity are intrinsic features of images. However, its application to other datasets where the position of the features is purely random creates a problem since the importance of locality is not satisfied. However, even in these cases, CNN has proven to be incredibly useful [22,23].

To overcome the problem of random positioned features for datasets where the location of the features does not provide any information, the solution has been to add successive convolutional layers, so that the deeper layers can integrate more distant features regardless of their spatial (not meaningful) location. In our case, the addition of a large number of convolutional layers is not feasible since we are facing a problem that is extremely prone to overfitting. That is the reason for proposing an alternative solution where, instead of a single input, we create k copies of the input, each of them formed with the same features but with their positions permuted. The permutation performed for each input is set randomly. It is important to note that the randomization of the features location is applied independently for each of the k inputs of the model, and that the random permutation of the features, once defined, is maintained throughout the training and prediction phases i.e. the position for each feature in each of the k inputs is different and randomly established, but once they are established their position is maintained, otherwise training and prediction would be completely stochastic and impossible.

In summary, creating k independent inputs with their features in permuted positions is one way to provide a CNN different feature layouts, not making it necessary to include a large number of layers, and bringing together in the same receptive field two features that could be highly correlated with widely separated initial locations. An additional option to deal with non-image data using a CNN is to perform a mapping between the original (non-image) to an image-like data by transferring similarity between the original features (in the non-image data) to location distance between these features in the transformed data, as proposed in [41]. This is an interesting alternative, but with clear challenges, the first and most important being the necessary a-priori definition of similarity, location distance and feature co-location in the new dataset. In summary, with this approach there is a need to manually define an appropriate feature representation (feature engineering problem), which is precisely the main problem solved by a CNN i.e. automatic extraction of feature representation. Avoiding this contradiction was the main reason for not following this alternative in this work.

The creation of k randomly permuted copies of the input comes along with two additional elements:

- A final averaging layer, that serves two purposes: 1) an additional ensemble property to prevent overfitting, and, 2) a way to obtain stable results considering the initial random allocation of features position.
- Weights shared among all sub-models, which reduces the amount of weights, decreasing the possibility of overfitting and creating a smoothing effect on the training level by imposing an average on gradient updates for all k inputs.

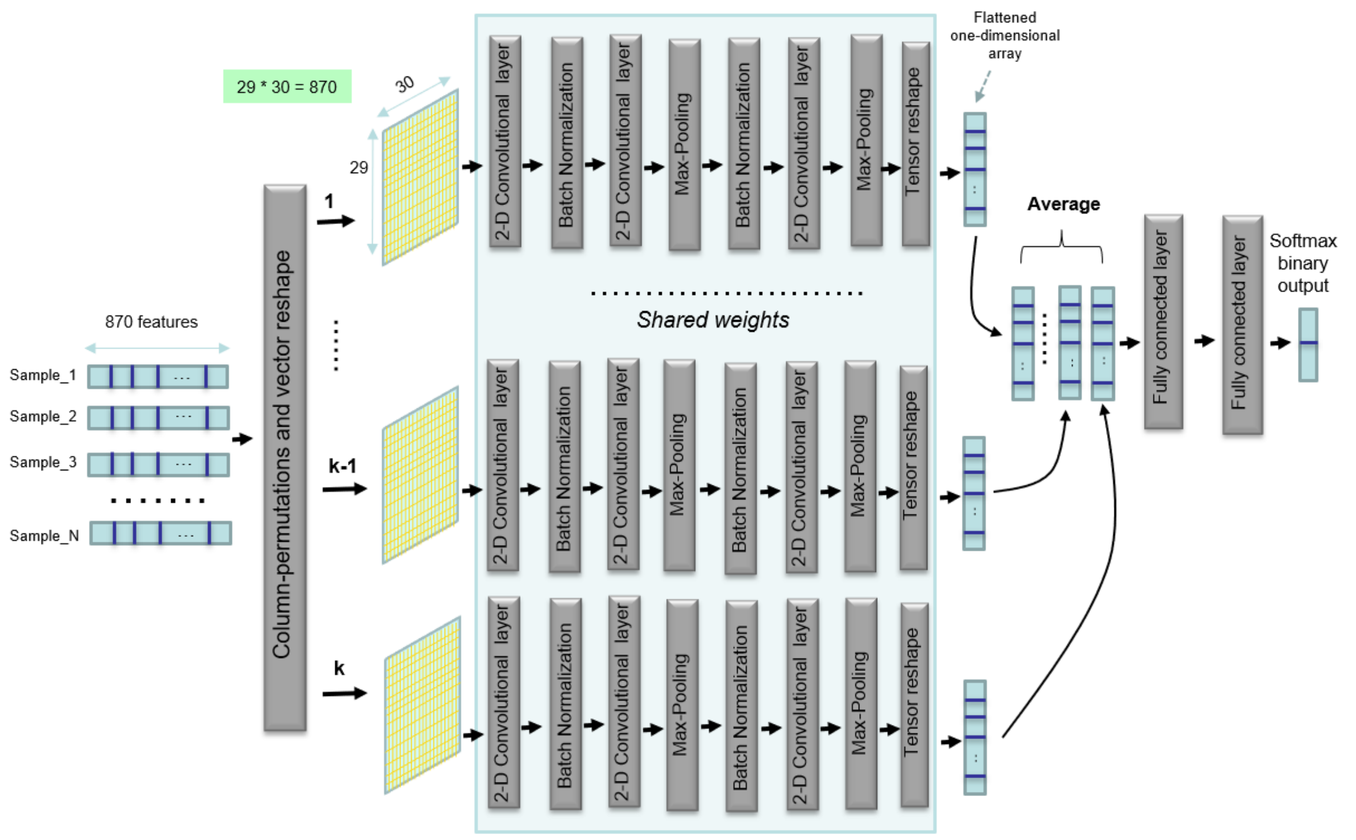


Fig. 5. Randomized 2-D CNN model (proposed model)

We implemented all the neural network models in python using Tensorflow/Keras. For all the classic ML models we used the scikit-learn python package [42]. To perform feature selection, we used the R packages: caret [36] and Boruta [37].

3.3 Validation & Test strategy

The test strategy followed is a two-stage nested cross-validation (CV) approach. This approach provides a more complete analysis, as a distribution of values under various configurations of sets of samples used for training and testing is obtained. The test strategy is presented graphically in Fig. 6. The inner-CV cycle is used to decide when to stop training, Meanwhile, the outer-CV cycle is used to randomize the training samples employed in the inner-CV. This allows having independent and randomly chosen test samples to produce the final test results. To increase the number of final results, the entire test architecture presented in Fig. 6 is repeated n times. Table II presents the specific values for the number of repetitions of the entire process (first numeric column), and the number of outer and inner iterations of the hierarchical CV strategy for each of the models. The total number of final test results obtained for each model is the product of the number of repetitions (Table II: first numeric column) and the number of outer iterations (Table II: second numeric column). The total number of test results corresponds to the "count" value in Fig. 7-8, which may be different for each model. These test results are those used to obtain the statistics per model presented in Fig. 7-8 (mean, standard deviation, quartiles...). Considering the large number of validation cycles and iterations and their impact on computing time, we chose to obtain more test results for the models with the best performance. This is why the number of results for the 2D-CNN and randomized 2D-CNN is 24 and 40, respectively. This number is 16 for the other models (Table II). The number of epochs and batch sizes used to train the models are also shown in Table II.

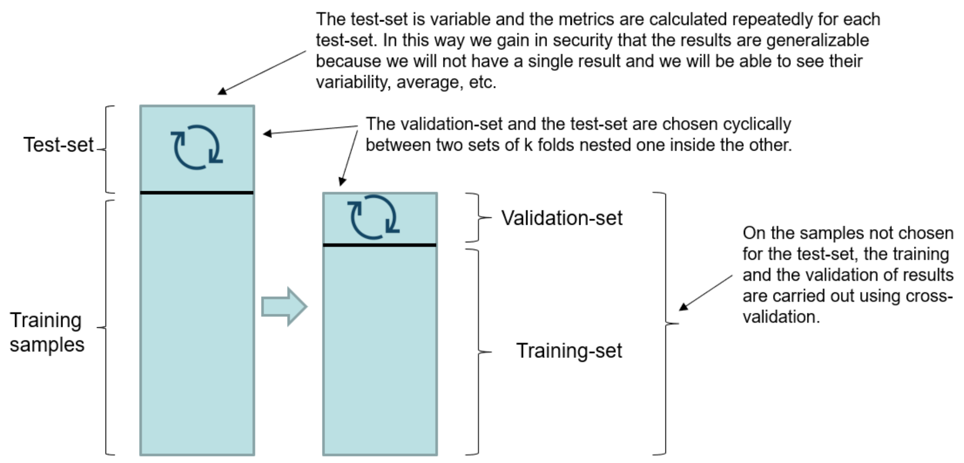

The test-set is variable and the metrics are calculated repeatedly for each test-set. In this way we gain in security that the results are generalizable because we will not have a single result and we will be able to see their variability, average, etc.
The validation-set and the test-set are chosen cyclically between two sets of k folds nested one inside the other.
Test-set
Training samples
Validation-set
Training-set
On the samples not chosen for the test-set, the training and the validation of results are carried out using cross-validation.

Fig. 6. Test strategy details. Several test sets are considered to obtain a distribution of classification performance metrics from which statistics such as the mean and standard deviation can be obtained.

Model	Repetitions	Outer iterations	Inner iterations	Epochs	Batch size
Randomized 2D-CNN	5	8	20	30	10
2D-CNN	3	8	50	80	10
1D-CNN	2	8	50	80	10
1D-CNN+GPC	2	8	50	80	10
Multilayer Perceptron	5	8	20	30	10
Extreme Learning Machine	5	8	20	30	10
Classic ML models	2	8	NA	NA	NA

Table II. Parameters used to implement the test strategy for the different models

4. RESULTS

In this section we present the classification performance results obtained with all the proposed models. The focus of this work was to investigate the suitability of employing several CNN architectures for the early detection of Alzheimer's disease, with an emphasis on proposing novel models that can be trained in the context of a small number of high-dimensional noisy signals. Here we show that our proposed model (randomized 2D-CNN) provides the best classification results. Special care was taken to the test strategy, to guarantee confidence in the results.

The results presented in this section come from applying the dataset described in Section 3.1 to various types of models:1) classic ML models: LR, NB, GBM, RF, GP Classifier and SVM with radial basis functions (RBF); 2) CNN architectures: 1D-CNN, 1D-CNN plus GPC, 2D-CNN and Randomized 2-D CNN models, and; 3) other neural network architectures: Multilayer Perceptron (MLP) and Extreme Learning Machine (ELM)[43].

An exhaustive comparison of the results is provided for all models (Fig. 7-8) with carefully selected classification metrics. Considering the small number of samples, it is especially important to provide the distribution of the performance metrics. For this reason, the mean, standard deviation, maximum and minimum values and the 25%, 50% (median) and 75% quartiles are provided for all performance metrics and all models. The selected classification performance metrics were: accuracy, F1-score, precision and recall. These four metrics provide a good overview of the different aspects and objectives of a classifier.

To define these metrics, and using the usual terminology, we consider the presence of MCI as a 'positive' result and the healthy state a 'negative' result. With these definitions, a true positive (TP) is a prediction that the patient suffers from MCI when that is actually the case. A true negative (TN) is defined as predicting that an actual healthy participant does not suffer from MCI. A false positive (FP) occurs when an actual healthy participant is classified as suffering from MCI. Finally, a false negative (FN) is defined as the instance when an actual patient is classified as not suffering from the condition. With these definitions, the metrics are expressed as:

$$Accuracy = \frac{\#TP+\#TN}{\#TP+\#TN+\#FP+\#FN} \quad (4)$$

$$Precision = \frac{\#TP}{\#TP+\#FP} \quad (5)$$

$$Recall = \frac{\#TP}{\#TP+\#FN} \quad (6)$$

$$F1-score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (7)$$

Where a '#' sign before a symbol denotes the total number of occurrences in that category. E.g., $\#TN$ is the total number of true negatives.

All the results presented are based on the dataset described in section 3.1, where the feature set FS2 (870 features) was used for the 2D-CNN and Randomized 2D-CNN models, and the feature set FS1(20 features) was used for all the other models.

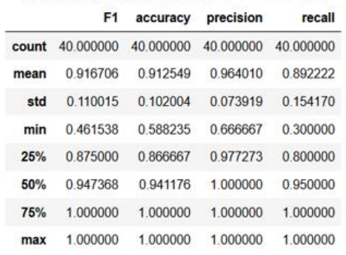
Fig. 7-8 provide separate results for all models with one table per model. There are four scores: accuracy, F1, precision and recall, and their associated statistics: mean, standard deviation, minimum and maximum values, three quantiles and the number of results used to perform the statistics (first row named "count"). A box-plot for each particular set of results is also provided below each model. The total number of test results corresponds to the "count" value in Fig. 7-8, which may be different for each model.

The recall metric is particularly important since it reflects the number of false positives, actual patients who have not been diagnosed, which is a number one would like to keep as low as possible. The precision metric provides an indication of the number of false positives, which should also be kept low, as diagnosing a healthy participant with the condition also has costs. This is the reason why the F1-score, which is the geometric mean of precision and recall, has been adopted as our main indicator for the prediction quality of the classifier. Fig. 7 shows that our proposed model (randomized 2D-CNN) provides the best values for the F1-score. It is superior with respect to all other metrics as well, except for the 1D-CNN model, that provides a better average result for the recall metric, but a worse median result for this metric compared to the randomized 2D-CNN model.

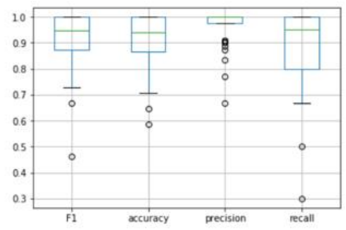
The reason for choosing the other 3 deep learning models (2D-CNN, 1D-CNN and 1D-CNN+GPC) has been to compare the proposed model with other simpler ones, which were also based on a convolutional neural network architecture. In our case, and given the longitudinal character of the original features, it was natural to use a 1D-CNN as one of the alternatives. A 2D-CNN model was also a clear candidate since our proposed model is based on it. Finally, the reason for adding a Gaussian Process Classifier (GPC) to the 2D-CNN was to investigate whether incorporating a non-parametric bayesian model could improve the results considering its excellent performance on problems with a limited number of samples. The other two neural network architectures (MLP and ELM) were chosen, since MLP is the basis for all neural network models and ELM was used for similar detection of MCI in previous works [27]. Finally, the classic ML models (LR, NB, GBM, RF, GP Classifier and SVM) were chosen to compare the results with well-known methods.

The classic ML classifiers (Fig. 8) are setup with a minimum hyper-parameters tuning from their default values in scikit-learn [42]. The Multilayer Perceptron (MLP) is formed by two hidden layers of 15 and 2 neurons with ReLU and softmax as activation functions for the middle and last layers and cross entropy as the loss function. The Extreme Learning Machine consists of a single hidden layer of 20 neurons with random initialization and sigmoid activations. The weights of the hidden layer are set as non-trainable, and the layer operates as a random projection. The end layer has a softmax activation with a cross entropy loss. The entire model is trained end-to-end like a normal neural network. The end layer functions similarly to a logistic regression and its weights are trained as proposed in [43], but without requiring additional least squares optimization.

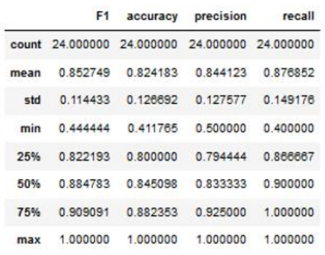
Randomized 2D-CNN



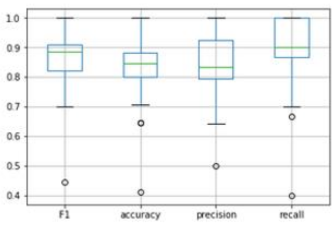
	F1	accuracy	precision	recall
count	40.000000	40.000000	40.000000	40.000000
mean	0.916706	0.912549	0.964010	0.892222
std	0.110015	0.102004	0.073919	0.154170
min	0.461538	0.588235	0.666667	0.300000
25%	0.875000	0.866667	0.977273	0.800000
50%	0.947368	0.941176	1.000000	0.950000
75%	1.000000	1.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000



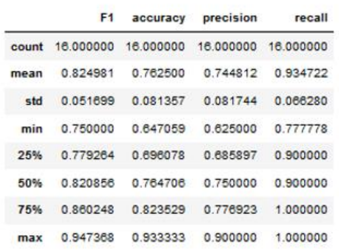
2D-CNN



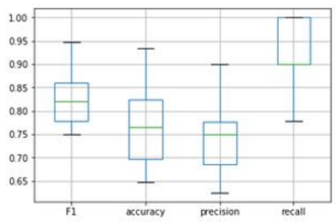
	F1	accuracy	precision	recall
count	24.000000	24.000000	24.000000	24.000000
mean	0.852749	0.824183	0.844123	0.876852
std	0.114433	0.126692	0.127577	0.149176
min	0.444444	0.411765	0.500000	0.400000
25%	0.822193	0.800000	0.794444	0.866667
50%	0.884783	0.845098	0.833333	0.900000
75%	0.909091	0.882353	0.925000	1.000000
max	1.000000	1.000000	1.000000	1.000000



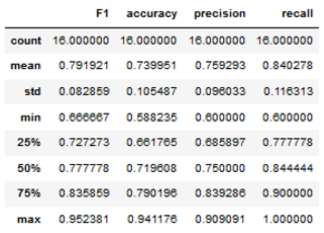
1D-CNN



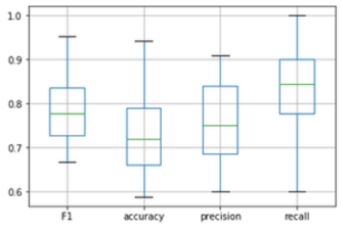
	F1	accuracy	precision	recall
count	16.000000	16.000000	16.000000	16.000000
mean	0.824981	0.762500	0.744812	0.934722
std	0.051699	0.081357	0.081744	0.066280
min	0.750000	0.647059	0.625000	0.777778
25%	0.779264	0.696078	0.685897	0.900000
50%	0.820856	0.764706	0.750000	0.900000
75%	0.860248	0.823529	0.776923	1.000000
max	0.947368	0.933333	0.900000	1.000000



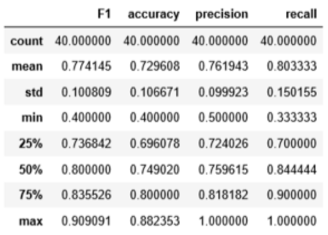
1D-CNN+GPC



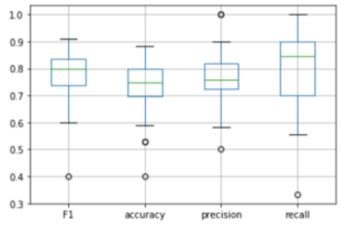
	F1	accuracy	precision	recall
count	16.000000	16.000000	16.000000	16.000000
mean	0.791921	0.739951	0.759293	0.840278
std	0.082859	0.105487	0.096033	0.116313
min	0.666667	0.588235	0.600000	0.600000
25%	0.727273	0.661765	0.685897	0.777778
50%	0.777778	0.719608	0.750000	0.844444
75%	0.835859	0.790196	0.839286	0.900000
max	0.952381	0.941176	0.909091	1.000000



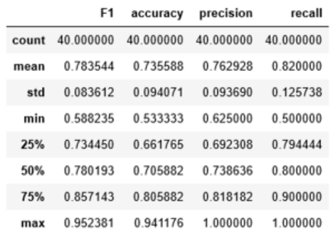
MLP



	F1	accuracy	precision	recall
count	40.000000	40.000000	40.000000	40.000000
mean	0.774145	0.729608	0.761943	0.803333
std	0.100809	0.106671	0.099923	0.150155
min	0.400000	0.400000	0.500000	0.333333
25%	0.736842	0.696078	0.724026	0.700000
50%	0.800000	0.749020	0.759615	0.844444
75%	0.835526	0.800000	0.818182	0.900000
max	0.909091	0.882353	1.000000	1.000000



ELM



	F1	accuracy	precision	recall
count	40.000000	40.000000	40.000000	40.000000
mean	0.783544	0.735588	0.762928	0.820000
std	0.083612	0.094071	0.093690	0.125738
min	0.588235	0.533333	0.625000	0.500000
25%	0.734450	0.661765	0.692308	0.794444
50%	0.780193	0.705882	0.738636	0.800000
75%	0.857143	0.805882	0.818182	0.900000
max	0.952381	0.941176	1.000000	1.000000

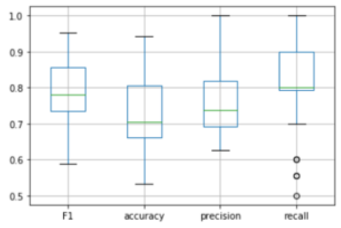
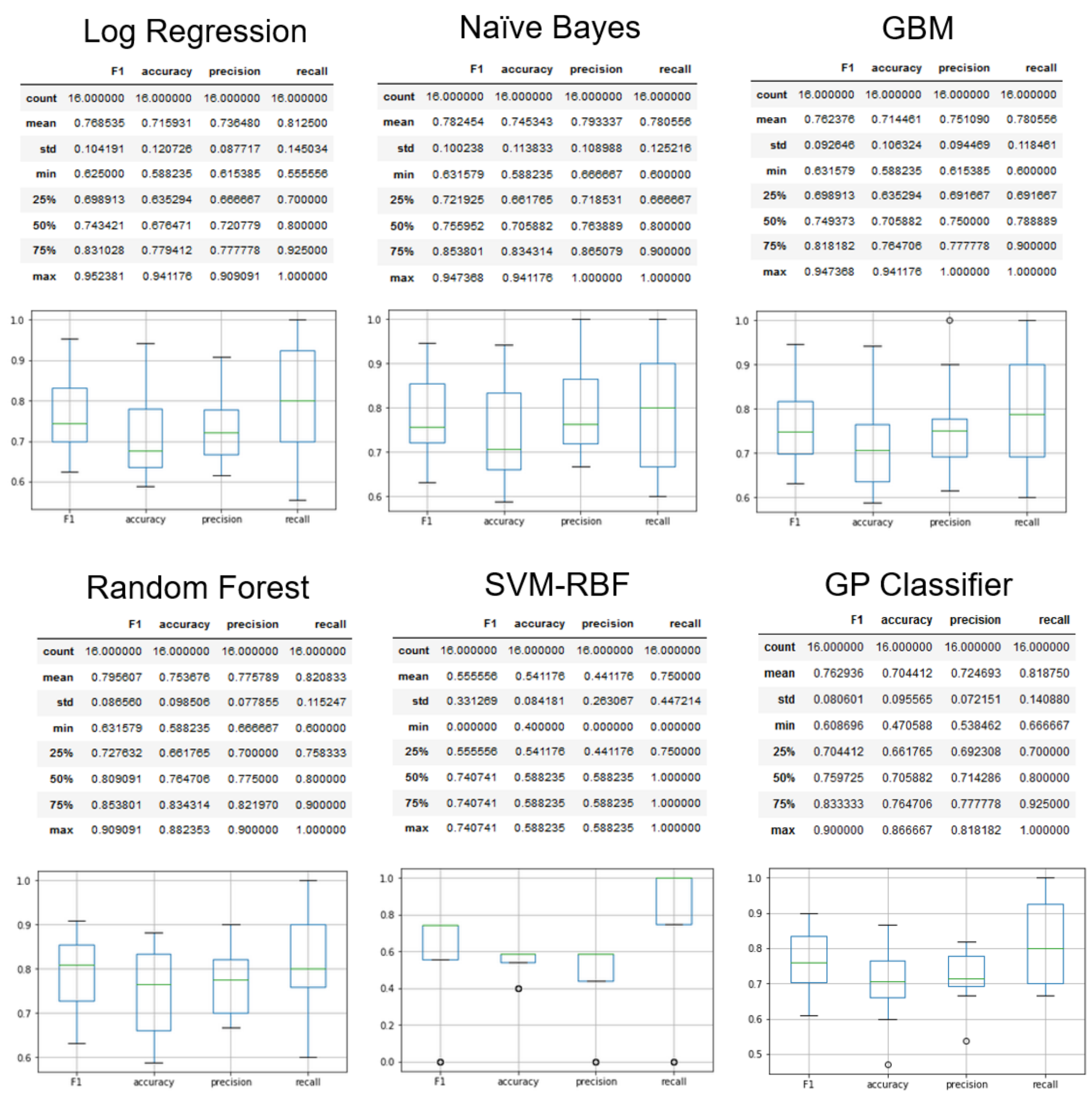


Fig. 7. Results obtained for all models based on neural networks



Log Regression

	F1	accuracy	precision	recall
count	16.000000	16.000000	16.000000	16.000000
mean	0.768535	0.715931	0.736480	0.812500
std	0.104191	0.120726	0.087717	0.145034
min	0.625000	0.588235	0.615385	0.555556
25%	0.698913	0.635294	0.666667	0.700000
50%	0.743421	0.676471	0.720779	0.800000
75%	0.831028	0.779412	0.777778	0.925000
max	0.952381	0.941176	0.909091	1.000000

Naïve Bayes

	F1	accuracy	precision	recall
count	16.000000	16.000000	16.000000	16.000000
mean	0.782454	0.745343	0.793337	0.780556
std	0.100238	0.113833	0.108988	0.125216
min	0.631579	0.588235	0.666667	0.600000
25%	0.721925	0.661765	0.718531	0.666667
50%	0.755952	0.705882	0.763889	0.800000
75%	0.853801	0.834314	0.865079	0.900000
max	0.947368	0.941176	1.000000	1.000000

GBM

	F1	accuracy	precision	recall
count	16.000000	16.000000	16.000000	16.000000
mean	0.762376	0.714461	0.751090	0.780556
std	0.092646	0.106324	0.094469	0.118461
min	0.631579	0.588235	0.615385	0.600000
25%	0.698913	0.635294	0.691667	0.691667
50%	0.749373	0.705882	0.750000	0.788889
75%	0.818182	0.764706	0.777778	0.900000
max	0.947368	0.941176	1.000000	1.000000

Random Forest

	F1	accuracy	precision	recall
count	16.000000	16.000000	16.000000	16.000000
mean	0.795607	0.753676	0.775789	0.820833
std	0.086560	0.098506	0.077855	0.115247
min	0.631579	0.588235	0.666667	0.600000
25%	0.727632	0.661765	0.700000	0.758333
50%	0.809091	0.764706	0.775000	0.800000
75%	0.853801	0.834314	0.821970	0.900000
max	0.909091	0.882353	0.900000	1.000000

SVM-RBF

	F1	accuracy	precision	recall
count	16.000000	16.000000	16.000000	16.000000
mean	0.555556	0.541176	0.441176	0.750000
std	0.331269	0.084181	0.263067	0.447214
min	0.000000	0.400000	0.000000	0.000000
25%	0.555556	0.541176	0.441176	0.750000
50%	0.740741	0.588235	0.588235	1.000000
75%	0.740741	0.588235	0.588235	1.000000
max	0.740741	0.588235	0.588235	1.000000

GP Classifier

	F1	accuracy	precision	recall
count	16.000000	16.000000	16.000000	16.000000
mean	0.762936	0.704412	0.724693	0.818750
std	0.080601	0.095565	0.072151	0.140880
min	0.608696	0.470588	0.538462	0.666667
25%	0.704412	0.661765	0.692308	0.700000
50%	0.759725	0.705882	0.714286	0.800000
75%	0.833333	0.764706	0.777778	0.925000
max	0.900000	0.866667	0.818182	1.000000

Fig. 8. Results obtained for all models based on classic ML models

Considering the interest of the results from the 2D-CNN randomized model, Fig. 9 provides a detailed set of histograms for the four statistics of results for this model (Fig. 9, upper part). Together with the histograms, Fig. 9 also offers an approximation to the probability density function for these four statistics (Fig. 9, bottom). It is interesting to appreciate that no more than 4 values (10% of the total values) are below 0.8 for any statistic, that is, 90% of the values obtained for any statistic are larger than 0.8, with most of them being much closer to 1.

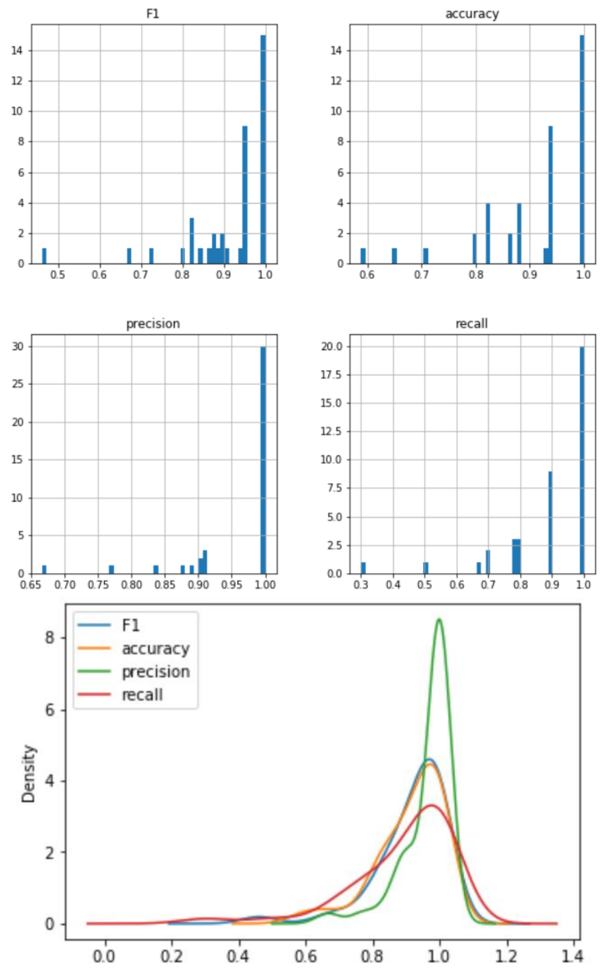
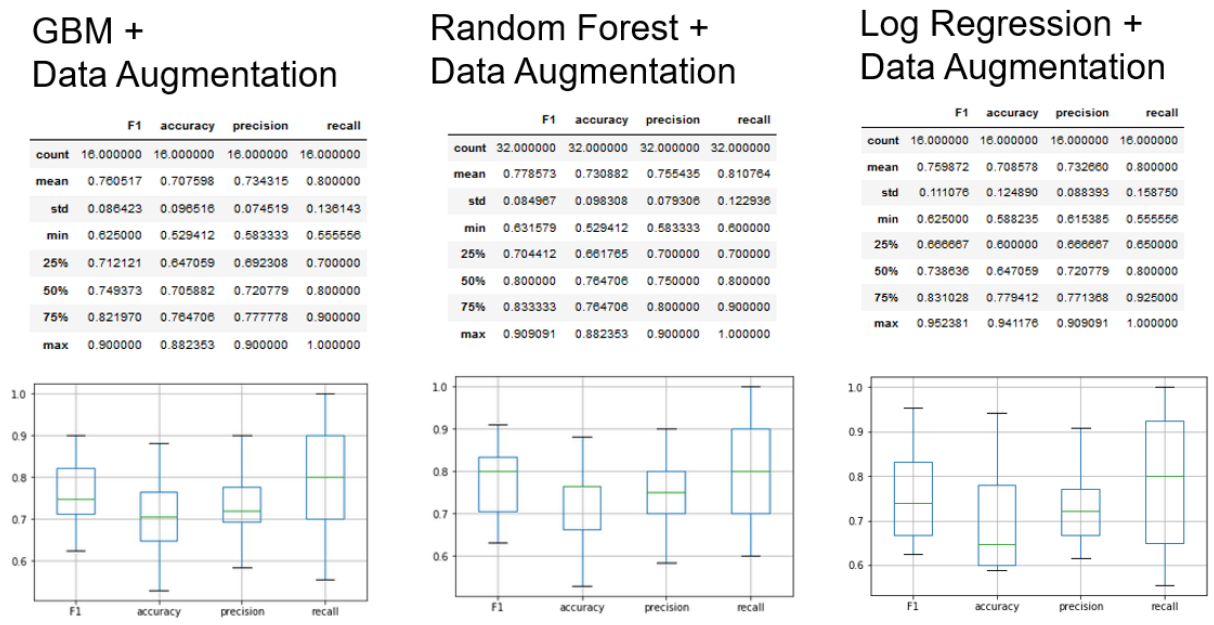


Fig. 9. Histograms (upper charts) and probability density function approximation (lower chart) for the results of the Randomized 2D-CNN model.

Given the small number of samples available for training, we additionally explored whether increasing the number of samples by data augmentation would be of benefit. Fig. 10 provides the results obtained by several models after a 3-fold increase in the number of samples. New samples were created by adding Gaussian noise with small variance to the original features. We can observe from the results in Fig. 10 that there is no improvement for any of the algorithms. We have not considered the use of methods such as the Synthetic Minority Over-sampling Technique (SMOTE) and its variants [44] since these methods designed for unbalanced datasets, which is not the case here. Additionally, the behavior of these methods when applied to high-dimensional data (large number of features) [45], which corresponds to the present situation, has not been fully investigated.



GBM + Data Augmentation

	F1	accuracy	precision	recall
count	16.000000	16.000000	16.000000	16.000000
mean	0.760517	0.707598	0.734315	0.800000
std	0.086423	0.096516	0.074519	0.136143
min	0.625000	0.529412	0.583333	0.555556
25%	0.712121	0.647059	0.692308	0.700000
50%	0.749373	0.705882	0.720779	0.800000
75%	0.821970	0.764706	0.777778	0.900000
max	0.900000	0.882353	0.900000	1.000000

Random Forest + Data Augmentation

	F1	accuracy	precision	recall
count	32.000000	32.000000	32.000000	32.000000
mean	0.778573	0.730882	0.755435	0.810764
std	0.084967	0.098308	0.079306	0.122936
min	0.631579	0.529412	0.583333	0.600000
25%	0.704412	0.661765	0.700000	0.700000
50%	0.800000	0.764706	0.750000	0.800000
75%	0.833333	0.764706	0.800000	0.900000
max	0.909091	0.882353	0.900000	1.000000

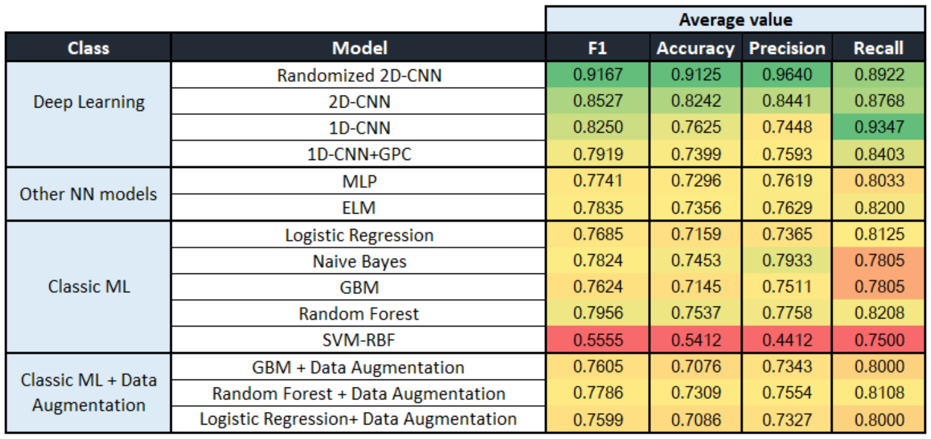
Log Regression + Data Augmentation

	F1	accuracy	precision	recall
count	16.000000	16.000000	16.000000	16.000000
mean	0.759872	0.708578	0.732660	0.800000
std	0.111076	0.124890	0.088393	0.158750
min	0.625000	0.588235	0.615385	0.555556
25%	0.666667	0.600000	0.666667	0.650000
50%	0.738636	0.647059	0.720779	0.800000
75%	0.831028	0.779412	0.771368	0.925000
max	0.952381	0.941176	0.909091	1.000000

Fig. 10. Results obtained using data augmentation

Furthermore, we built ensembles of models based on various combinations of models: (1) NB, GBM and RF, and (2) NB, GBM, RF and GPC; using a majority voting strategy to select the final label. This ensemble approach did not provide any improvement compared to the best individual model used to conform the ensemble.

Figs. 11 and 12 summarize all the results presented previously. Fig. 11 provides the average (mean) values for all metric scores and all models. The figure is color-coded column-wise, with a color palette from green to red, where the greenest color is for best results and the reddest for worst results. Given the importance of the F1-score, the bars in Fig. 12 show the corresponding average F1 value for all models. Looking at Figs. 11 and 12, it can be seen how the Randomized 2D-CNN model is the one that provides the best classification results.



		Average value			
Class	**Model**	**F1**	**Accuracy**	**Precision**	**Recall**
Deep Learning	Randomized 2D-CNN	0.9167	0.9125	0.9640	0.8922
	2D-CNN	0.8527	0.8242	0.8441	0.8768
	1D-CNN	0.8250	0.7625	0.7448	0.9347
	1D-CNN+GPC	0.7919	0.7399	0.7593	0.8403
Other NN models	MLP	0.7741	0.7296	0.7619	0.8033
	ELM	0.7835	0.7356	0.7629	0.8200
Classic ML	Logistic Regression	0.7685	0.7159	0.7365	0.8125
	Naive Bayes	0.7824	0.7453	0.7933	0.7805
	GBM	0.7624	0.7145	0.7511	0.7805
	Random Forest	0.7956	0.7537	0.7758	0.8208
	SVM-RBF	0.5555	0.5412	0.4412	0.7500
Classic ML + Data Augmentation	GBM + Data Augmentation	0.7605	0.7076	0.7343	0.8000
	Random Forest + Data Augmentation	0.7786	0.7309	0.7554	0.8108
	Logistic Regression+ Data Augmentation	0.7599	0.7086	0.7327	0.8000

Fig. 11. Average values of all performance metrics for all models evaluated

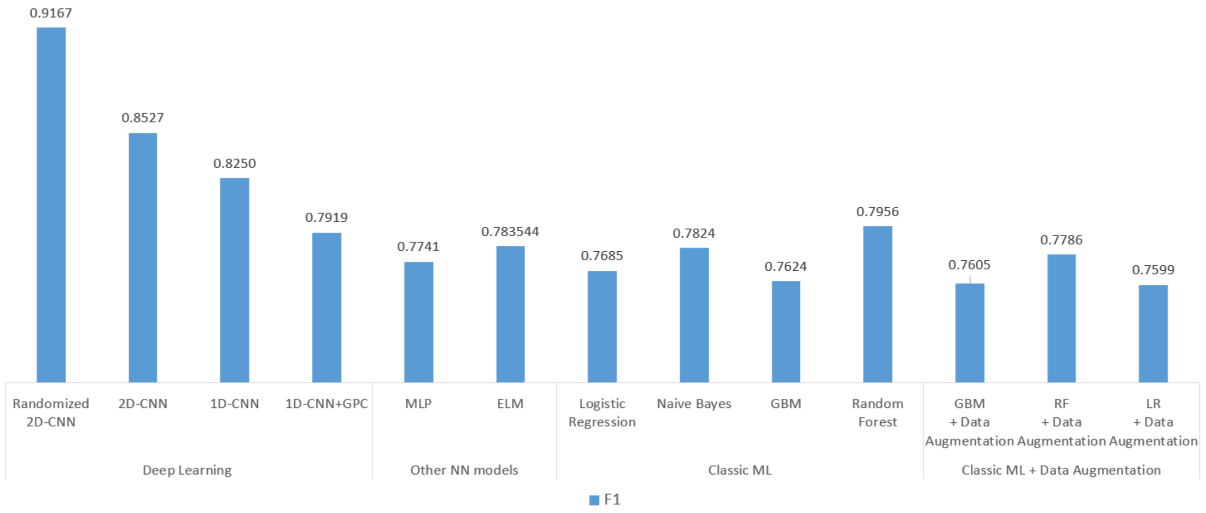
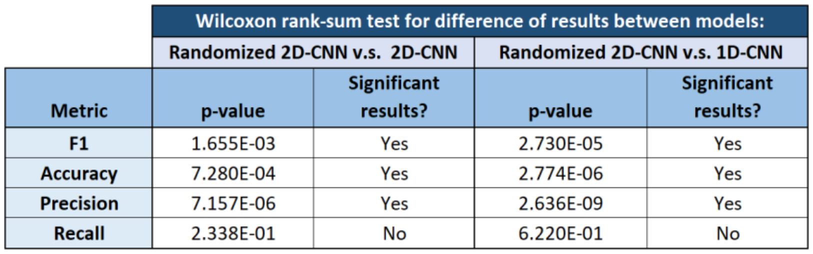


Fig. 12. Detailed diagram of the average values of F1-scores for all models evaluated

To ensure that the good results obtained by the Randomized 2D-CNN model are significantly better than those obtained by other models from a statistical point of view, Fig.13 presents the application of the Wilcoxon one-sided unpaired rank-sum test for the comparison of performance metrics between the best model (Randomized 2D-CNN) and the next two best (2D-CNN and 1D-CNN). The Wilcoxon test is a nonparametric test that checks whether two groups of values show differences in their population mean ranks. It does not assume any a-priori probability distribution of the values. The Wilcoxon test was applied to compare the results of the best model with each of the other two models separately. Fig.13 presents these two separate tests with two independent blocks of results (left and right parts of Fig.13), each providing the p-value resulting from the test and the test results that allow (or not) to reject the null hypothesis that is associated with a non-significant difference in the results. The test used was one-sided to specifically check if one of the groups has higher ranked mean that the other. The values in Fig.13 are adjusted for multiple comparisons correction (Bonferroni)[46]. The significance level used was 1%. From Fig.13 we can conclude that the results achieved by the best model are significantly better than the others in three of the four metrics. Only the recall metric presents a non-significant result at a level of significance of 1%, even when the best model has the highest median value for recall (Fig. 7); the greater variance of the recall results (Fig. 9) could be an explanation for this result.



	Wilcoxon rank-sum test for difference of results between models:			
	Randomized 2D-CNN v.s. 2D-CNN		Randomized 2D-CNN v.s. 1D-CNN	
Metric	p-value	Significant results?	p-value	Significant results?
F1	1.655E-03	Yes	2.730E-05	Yes
Accuracy	7.280E-04	Yes	2.774E-06	Yes
Precision	7.157E-06	Yes	2.636E-09	Yes
Recall	2.338E-01	No	6.220E-01	No

Fig.13. Wilcoxon rank-sum test. This test checks if the results for the best model (Randomized 2D-CNN) are statistically significantly better compared to each of the next two best models: 2D-CNN (left sub-table) and 1D-CNN (right sub-table)

5. Conclusion

This work presents a novel deep learning architecture based on CNN networks for the detection of Mild Cognitive Impairment. The proposed model applies stochastic permutation of the training data (synchronization measures extracted from MEG signals of the patients during rest) employing an ensemble of identical learning blocks formed by a combination of CNN, batch-normalization and max-pooling layers. The outputs from the ensemble are aggregated (averaged) and constitute the prediction output of the classifier. We show that the combination of (1) CNN networks, (2) the averaging capabilities of the ensemble structure and (3) the exploration of new feature interactions (provided by their random permutation), creates a final architecture that offers excellent classification results for a challenging scenario characterized by sample scarcity and a high-dimensional feature space.

We provide empirical results that show that the proposed model offers significantly better classification metrics than alternative models based on 2D-CNN and 1D-CNN networks and classic machine learning models (logistic regression, random forest, GBM ...). To make the comparison, we apply a hierarchical cross-validation strategy that produces several measurements (based on different test-sets) of the classification metrics, whose statistics reinforce confidence in the generalization of the outcomes. A complete and detailed analysis of the results is provided.

The dataset used was real MEG data obtained at the Centre for Biomedical Technology, (Madrid, Spain) from 78 Mild Cognitive Impairment (MCI) patients and 54 healthy aging participants. The signals post-processing and data preparation necessary to train the algorithms is presented in detail.

This work shows that it is possible to apply CNN models to scenarios that are highly prone to overfitting with an appropriate architecture based on (1) reducing the number of parameters as much as possible, (2) using an ensemble of similar structures that share their weights, (3) averaging the results of the ensemble and (4) the permutation of entries to create new input data randomly altered and with different co-location of features. Given the good prediction results of the proposed model, this approach could be the basis for practical applications in the early detection of Alzheimer’s disease based on MEG activity.

In the present work, the features that were used as input to the classifier were measures of synchronization between MEG channel signals. There are other types of variables such as demographic variables like age and sex, and neuropsychological variables like the MMSE scores that can also contribute to diagnosing patients. It is therefore very likely that including some of these variables as input to the classifier would improve the classification performance. This is a possible avenue for future work. On the other hand, while classification was the focus of this work, the fact that brain activity can serve as a basis for diagnosis indicates that the two diagnostic groups show different brain activity profiles, which is of interest from a fundamental point of view. In that context, to the extent that the two groups differ in demographic or neuropsychological values, the identified difference in brain activity profiles may partially reflect differences in these other variables rather than differences in diagnosis. There were some differences in the age distribution, as well as in the proportion of females and males, between patients and controls. These between-group age differences are also likely to be found in a clinical setting. On the other hand, MMSE scores probably require a somewhat different consideration in the present discussion. While the relationship between MMSE scores and diagnosis is not one to one, it is significant [33]. It would therefore not be possible, and probably not desirable, to define groups matched for MMSE score. Nevertheless, at a fundamental level, it could be interesting to investigate in future works to what extent between-group differences in brain activity are more closely related to MMSE scores or diagnosis. Future work could also investigate to what extent the methods described here are robust when data from different scanners is used for diagnosis; future research could also focus on the combination of deep learning models and stochastic methods.

This work is also a proof of concept to assess the suitability of randomized input 2D-CNN models to make predictions on noisy and small datasets with a large number of features. And, it opens up interesting research opportunities for other areas with similar problems e.g. network intrusion detection... This allows us to validate the algorithms and methods developed from previous research activities on network traffic analysis and prediction, extending the scope of our research to other fields such as neuroscience.

ACKNOWLEDGEMENTS

The collection of the data employed in the analysis was partially funded with grant PSI2015-68793-C3-1-R from the Spanish Ministry for Science, Innovation and Universities.

The preparation of the article and the study of algorithms were partially funded with grant RTI2018-098958-B-I00 from Proyectos de I+D+i «Retos investigación», Programa Estatal de I+D+i Orientada a los Retos de la Sociedad, Plan Estatal de Investigación Científica, Técnica y de Innovación 2017-2020. Spanish Ministry for Science, Innovation and Universities.

REFERENCES

[1] Khan A. and Usman M., "Early diagnosis of Alzheimer's disease using machine learning techniques: A review paper," 2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K), Lisbon, 2015, pp. 380-387. https://doi.org/10.5220/0005615203800387

[2] Ullah H.M.T. et al. "Alzheimer's Disease and Dementia Detection from 3D Brain MRI Data Using Deep Convolutional Neural Networks," 2018 3rd International Conference for Convergence in Technology (I2CT), Pune, 2018, pp. 1-3. https://doi.org/10.1109/I2CT.2018.8529808

[3] Petersen R.C. “Mild cognitive impairment”. New England Journal of Medicine, 2011, 364(23), 2227-2234. https://doi.org/10.1056/NEJMcp0910237

[4] Robertson K. et al. “Using Varying Diagnostic Criteria to Examine Mild Cognitive Impairment Prevalence and Predict Dementia Incidence in a Community-Based Sample”. Journal of Alzheimer’s Disease. 2019;68(4):1439-1451. https://doi.org/10.3233/JAD-180746

[5] López-Sanz D., Serrano N. and Maestú F. “The Role of Magnetoencephalography in the Early Stages of Alzheimer's Disease”. Frontiers in Neuroscience. 2018;12:572. https://doi.org/10.3389/fnins.2018.00572

[6] Mandal P.K. et al. “A Comprehensive Review of Magnetoencephalography (MEG) Studies for Brain Functionality in Healthy Aging and Alzheimer's Disease (AD).” Frontiers in computational neuroscience. Vol. 12 60. 2018, https://doi.org/10.3389/fncom.2018.00060

[7] Maestú F. et al. “A multicenter study of the early detection of synaptic dysfunction in Mild Cognitive Impairment using Magnetoencephalography-derived functional Connectivity”. 2015.. Neuroimage Clinical. 9, 103–109. https://doi.org/10.1016/j.nicl.2015.07.011

[8] Bi XA. et al. “Analysis of Alzheimer's Disease Based on the Random Neural Network Cluster in fMRI”. Frontiers in Neuroinformatics. 2018 Sep 7;12:60. https://doi.org/10.3389/fninf.2018.00060

[9] Meszlenyi R., Buza K..and Vidnyanszky Z. “Resting state fMRI functional connectivity-based classification using a convolutional neural network architecture“. arXiv:1707.06682 [stat.ML]. 2017.

[10] Ruffini G. et al. “EEG-driven RNN classification for prognosis of neurodegeneration in at-risk patients”. Artificial Neural Networks and Machine Learning – ICANN 2016. Lecture Notes in Computer Science, vol 9886. Springer. https://doi.org/10.1007/978-3-319-44778-0_36

[11] Ruffini G. et al. “Deep learning using EEG spectrograms for prognosis in idiopathic rapid eye movement behavior disorder (RBD)”. bioRxiv 240267; 2018. https://doi.org/10.1101/240267

[12] Wu W., Nagarajan S. and Chen Z. "Bayesian Machine Learning: EEG\/MEG signal processing measurements," in IEEE Signal Processing Magazine, vol. 33, no. 1, pp. 14-36, 2016. https://doi.org/10.1109/MSP.2015.2481559

[13] Kostas D. Pang E.W and Rudzicz F. “Machine learning for MEG during speech tasks”. Nature. Scientific Reports 9. 1609 (2019) https://doi.org/10.1038/s41598-019-38612-9

[14] Martinez-Murcia F.J. et al. “A 3D Convolutional Neural Network Approach for the Diagnosis of Parkinson’s Disease”. Natural and Artificial Computation for Biomedicine and Neuroscience. IWINAC 2017. Lecture Notes in Computer Science, vol 10337. Springer. 2017.

[15] Aoe J. et al. “Automatic diagnosis of neurological diseases using MEG signals with a deep neural network”. 2019. Nature. Scientific Reports. 9, 5057 (2019). https://doi.org/10.1038/s41598-019-41500-x

[16] Cichy R.M. et al. “Dynamics of scene representations in the human brain revealed by magnetoencephalography and deep neural networks”, NeuroImage, Vol 153, 2017, pp. 346-358, https://doi.org/10.1016/j.neuroimage.2016.03.063

[17] Hasasneh A. et al. “Deep Learning Approach for Automatic Classification of Ocular and Cardiac Artifacts in MEG Data”. 2018. Hindawi. Journal of Engineering Vol 2018, Article ID 1350692. https://doi.org/10.1155/2018/1350692

[18] Zubarev I. et al. “Adaptive neural network classifier for decoding MEG signals”. NeuroImage, Vol 197, 2019, pp. 425-434, https://doi.org/10.1016/j.neuroimage.2019.04.068

[19] Rawat W. and Wang Z., “Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review” Neural Computation, vol. 29, no. 9, 2017, pp. 2352–449. https://doi.org/10.1162/neco_a_00990

[20] Ioffe S. and Szegedy C.”Batch normalization: Accelerating deep network training by reducing internal covariate shift". 2015. arXiv:1502.03167 [cs.LG]

[21] Lee Ch.-Y., Gallagher P.W., and Tu Z. “Generalizing pooling functions in convolutional neural networks: Mixed, Gated, and Tree'. 2015. arXiv:1509.08985 [stat.ML]

[22] Lopez-Martin M. et al., “Network traffic classifier with convolutional and recurrent neural networks for Internet of Things”, IEEE Access, vol. 5, pp. 18042-18050, 2017. https://doi.org/10.1109/ACCESS.2017.2747560

[23] Lopez-Martin M. et al., "Deep Learning Model for Multimedia Quality of Experience Prediction Based on Network Flow Packets," in IEEE Communications Magazine, vol. 56, no. 9, pp. 110-117, Sept. 2018. https://doi.org/10.1109/MCOM.2018.1701156

[24] Srivastava N. et al. “Dropout: a simple way to prevent neural networks from overfitting”. The Journal of Machine Learning Research. 15, 1, pp.1929-1958. 2014. http://jmlr.org/papers/v15/srivastava14a.html

[25] An G., "The Effects of Adding Noise During Backpropagation Training on a Generalization Performance," in Neural Computation, vol. 8, no. 3, pp. 643-674, 1996. https://doi.org/10.1162/neco.1996.8.3.643

[26] Bishop C.M., “Training with Noise is Equivalent to Tikhonov Regularization”. Neural Computation. 1995 7:1, 108-116. https://doi.org/10.1162/neco.1995.7.1.108

[27] Weiming L. et al. “Convolutional Neural Networks-Based MRI Image Analysis for the Alzheimer’s Disease Prediction From Mild Cognitive Impairment”. Frontiers in Neuroscience. Vol 12, 2018. https://doi.org/10.3389/fnins.2018.00777

[28] Zhang X. et al. “ShuffleNet: An Extremely Efficient Convolutional Neural Network for MobileDevices”. 2017. arXiv:1707.01083 [cs.CV]

[29] Brendel W. and Bethge M. “Approximating CNNs with Bag-of-local-Features models works surprisingly well on

ImageNet”. 2019. arXiv:1904.00760 [cs.CV]

[30] Juefei-Xu F., Boddeti V.N and Savvides M. “Perturbative Neural Networks”. 2018. arXiv:1806.01817 [cs.CV]

[31] Zhao G. et al. “Random Shifting for CNN: a Solution to Reduce Information Loss in Down-Sampling Layers”. Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17). 2017. pp 3476-3482. https://doi.org/10.24963/ijcai.2017/486

[32] Ivan C. “Convolutional Neural Networks on Randomized Data”. 2019. arXiv:1907.10935 [cs.CV]

[33] Petersen R.C. “Mild cognitive impairment as a diagnostic entity”. J Intern Med. 2004 Sep;256(3):183-94. Review. https://doi.org/10.1111/j.1365-2796.2004.01388.x

[34] Taulu S and Simola J. “Spatiotemporal signal space separation method for rejecting nearby interference in MEG measurements". Phys Med Biol. 2006 Apr 7;51(7):1759-68. https://doi.org/10.1088/0031-9155/51/7/008

[35] Oostenveld R, Fries P, Maris E and Schoffelen JM. “FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data”. Comput Intell Neurosci. 2011;2011:156869. https://doi.org/10.1155/2011/156869

[36] Kuhn M. “caret: Classification and Regression Training”. R package. 2019. https://cran.r-project.org/web/packages/caret/caret.pdf

[37] Kursa M.B. “Boruta: Wrapper Algorithm for All Relevant Feature Selection“. R package. 2018. https://cran.r-project.org/web/packages/Boruta/Boruta.pdf

[38] Ambroise, C. and McLachlan, G.J. “Selection bias in gene extraction on the basis of microarray gene-expression data”. Proceedings of the National Academy of Sciences, 2002, 99 (10) 6562-6566. https://doi.org/10.1073/pnas.102102699

[39] Kiranyaz S. et al. “1D Convolutional Neural Networks and Applications – A Survey”. 2019. arXiv:1905.03554 [eess.SP]

[40] Williams C.K.I. and Barber D., “Bayesian Classification with Gaussian Processes” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 20, no. 12, 1998, pp. 1342–51. https://doi.org/10.1109/34.735807

[41] Sharma, A. et al. “DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture”. Sci Rep 9, 11399 (2019). https://doi.org/10.1038/s41598-019-47765-6

[42] Pedregosa F. et al., “Scikit-learn: Machine Learning in Python”, Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[43] Huang, G.B., Zhu, Q.Y. and Siew, Ch. K., “Extreme learning machine: theory and applications". Neurocomputing. 70 (1): 489–501. 2006. https://doi.org/10.1016/j.neucom.2005.12.126.

[44] Batista G.E.A.P.A, Prati R.C., and Monard M.C. “A study of the behavior of several methods for balancing machine learning training data”. ACM SIGKDD Explorations Newsletter. 6(1):20–29. 2004. https://doi.org/10.1145/1007730.1007735

[45] Blagus R. and Lusa L. “SMOTE for high-dimensional class-imbalanced data”. BMC Bioinformatics. 14-106. 2013. https://doi.org/10.1186/1471-2105-14-106

[46] Chen, SY et al. “A general introduction to adjustment for multiple comparisons.” Journal of thoracic disease vol. 9,6 (2017): 1725-1729. https://doi.org/10.21037/jtd.2017.05.34